ABSTRACT

The SAS macro function %SYSFUNC is very useful because it makes possible the use of SAS data step functions in the SAS macro environment. However, when the argument in the nested SAS data step function contains a macro variable or another macro such as %sysfunc(substr(&&AB&C,1)), it is hard to interpret the result without an understanding of the functionalities of %SYSFUNC and the interaction of %SYSFUNC with the nested SAS functions. Through a series of tests, the authors explored the functionalities of %SYSFUNC and the working order of %SYSFUNC with the nested SAS data step functions. The macro variable in a data step function is resolved first, then the data step function starts to work on the resolved value of the macro variable, and finally, %SYSFUNC resolves macros or macro variables in the output of data step function - but only once.

INTRODUCTION

The SAS macro function %SYSFUNC facilitates the use of SAS data step functions in the SAS macro environment saving a lot of time and effort. However, when the argument in the nested SAS function contains macro variables or other macros, quite often questions are asked about the mechanism behind the combinations, for example:

“What is the working order of %SYSFUNC with the nested SAS data step functions? Is a macro variable resolved before or after it is fed into the SAS data step function? If the output from the data step function contains a macro variable, when is it resolved”?

“If the macro variable in the argument of a data step function is associated with another macro, how is it passed into the function? If the argument in the nested SAS function contains several macro variables or macros, can they be resolved to plain text”?

This paper will address the above questions through examples.

WORKING ORDER OF %SYSFUNC WITH NESTED SAS DATA STEP FUNCTIONS

1. If the argument of %SYSFUNC is a data step function and if the argument of the data step function has a macro variable, the macro variable is resolved first. Next, the resolved value is fed into the argument of SAS data step function. Finally, the data step function starts execution.

In the following example, macro variable A is resolved to string TEST first; then the string TEST is fed into a data step function SUBSTR; finally, the data step function executes. Therefore, the output is Y=TEST.

```
%let A=TEST;
%put Y = %sysfunc(substr(&A,1));
```
2. If the output of a data step function contains a macro variable, the macro variable is resolved when the output is nested within %SYSFUNC.

In the sample code below, macro variable X is assigned value &A; the output from the data step function SUBSTR is &A; %SYSFUNC resolves it to TEST:

```sas
%let A=TEST;
%let X=%nrstr(&A);
%put Y = %sysfunc(substr(&X,1));
```

** The resolved value of Y in SAS log is: Y = TEST;

3. Function %SYSFUNC resolves macro variables or macros after the execution of the data step functions and is illustrated in the following sample codes:

```sas
%let T=DATE();
%let Y=%nrstr(&T);
%put Z=%sysfunc(substr(&Y, 1, 2));
```

** The resolved value of Z in SAS log is: Z = DATE();

In the second line of the above sample code, macro variable Y is stored as &T, which is also the output from SUBSTR in the third line. The %SYSFUNC resolves the output from function substr(&Y, 1, 2), which is &T, to DATE(). This resolution is performed after the execution of the data step function SUBSTR. Therefore, the result from %sysfunc(substr(&Y, 1, 2)) is DATE() rather than DA.

The next example shows that %SYSFUNC resolves the macro variable in the end if the output from the data step function that is called by %SYSFUNC contains a macro variable.

The order of the process can be more clearly illustrated by the following more complicated example code:

```sas
%macro procA;
   &
%mend procA;
%macro procB;
D
%mend procB;

%let D = TEST;
%let C=%nrstr(%procB);
%let X=%str(%procA&C);
%put Y1 = %sysfunc(substr(&X, 1));
%put Y3 = %sysfunc(substr(&X, 3));
```

What can we realize from the following code?

```sas
%put Y1 = %sysfunc(substr(&X, 1));
%put Y3 = %sysfunc(substr(&X, 3));
```
Macro variable X is stored in macro symbol table as &%procB which is passed to SUBSTR. The output of substr(&X, 1), &%procB, is resolved by %SYSFUNC to &D which is the result of Y1. As the output from substr(&X, 3) is procB which contains no macro variable or macro, the result of Y3 is procB.

We can test the process further from the output as follows:

```sas
** In the code below, X = &%procB and the result of %procB is D;
%put Y11 = %sysfunc(substr(&X, 1, 1));
** The result from SUBSTR function is & and Y11 = &;
%put Y21 = %sysfunc(substr(&X, 1, 3));
** There is a message in log: WARNING: Apparent invocation of macro P not resolved.Y21 = &%p;
%put Y31 = %sysfunc(substr(&X, 1, 7));
** The result from SUBSTR function is &%procB and Y31 = &D;
```

**THE RESOLUTION OF MACRO VARIABLES IN THE NESTED SAS FUNCTIONS**

1. The values of macro variables in the arguments of a data step function depend on the directly stored value.

   If a stored macro variable is associated with another macro variable, the stored value of the macro variable in argument is fed into the data step function.

   For example, assign macro variable T to the value of DATE() and macro variable X to the value of &T, then the stored value of X is DATE(). The output from %sysfunc(&X, date7.) is the date of the execution date, say 04NOV07. If X is assigned as %nrstr(&T), the stored value of X is &T which is passed to %SYSFUNC. Since &T is a character string not a date, %SYSFUNC looks for a data step function instead. When %SYSFUNC fails to find a data step function in its argument, SAS issues an error message in the log file. We can see the difference from the code below:

```sas
%let T=DATE();
%let X=&T;
** The stored value of X is DATE();
%put %sysfunc(&X, date7.);
** The result is 04NOV07;
%let Z=%nrstr(&T);
** The stored value of Z is Z=&T;
%put %sysfunc(&Z, date7.);
** ERROR: Function name missing in %SYSFUNC or %QSYSFUNC macro function reference.
```

The code below clearly explains how and when a macro variable in a data step function called by %SYSFUNC is resolved.

```sas
%macro procA;
```
2. Function %SYSFUNC resolves the macro variable in the output of the data step function once, and not to the end of plain text.

Let us continue from the example above. If we reset X = %nrstr(%procA%procB), the output from statement substr(&X, 1) is %procA%procB, then %SYSFUNC resolves %procA and %procB to & and D respectively. The final result is &D, not the value of macro variable D, TEST.

%let X=%nrstr(%procA%procB);
**The stored value of X is %procA%procB;

%put Y = %sysfunc(substr(&X,1));
**The resolved value in SAS log is: Y = &D, not TEST;

Function %SYSFUNC resolves all macro variables no matter whether macro variables are masked with %NRBQUOTE or not.

Function %NRBQUOTE cannot stop %SYSFUNC from resolving the macro variables in data step function called by %SYSFUNC.

In the sample below, X = %procA%procB which has a value of TEST. The output from %sysfunc(substr(%nrquote(&X),1,2)) is TE not TEST as X is resolved as TEST before it is fed into SUBSTR.

%let X=%procA%procB;
** The stored value of X is X = TEST;

%put Y = %sysfunc(substr(%nrquote(&X),1,2));
** The resolved value in SAS log is: Y = TE;
3. Function %QSYSFUNC can prevent the macro variables in the output of data set function from being resolved. If there is a need to mask some characters or mnemonics, use %QSYSFUNC.

For example, if X is assigned %proA%proB, then output from statement %qsysfunc(substr(&X,1)) is %procA%procB.

```
%let X=%nrstr(%procA%procB);
%put Y = %qsysfunc(substr(&X,1));
** The resolved value in SAS log is: Y = %procA%procB;
```

CONCLUSIONS

From the examples above, we know that first the macro variable in a data step function is resolved according to the directly stored value of the macro variable. Function %SYSFUNC resolves macro variables or macros, if any, after the data step function executes. %SYSFUNC will resolve any macro variable or macro in the output of the data step function only once. When there is a need to mask some characters or mnemonics, %QSYSFUNC must be used.

ACKNOWLEDGEMENT

Authors thank the support from the Department of Scientific Programming Rahway Site, Merck Co., & Inc.

REFERENCES


CONTACT INFORMATION

Your comments and questions are valuable and appreciated. Authors can be reached at

Lin Yan
Merck & Co., Inc.
Rahway, NJ 07065 U.S.A.
Tel: 732-594-5949
Email: lin_yan2@merck.com

Helen Wang
Merck & Co., Inc.
Rahway, NJ 07065 U.S.A.
Email: Helen_wang@merck.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.